Software Abstractions for Simulation and Control of a Continuum Robot

Arne Nordmann, Matthias Rolf and Sebastian Wrede

Research Institute for Cognition and Robotics, Bielefeld University, Germany

Abstract. The Bionic Handling Assistant is a new continuum robot which is manufactured in a rapid-prototyping procedure out of elastic polyamide. Its mechanical flexibility and low weight provide an enormous potential for physical human robot interaction. Yet, the elasticity and parallel continuum actuation design challenge standard approaches to deal with a robot from a control, simulation, and software modeling perspective. We investigate how the software abstractions of the existing Robot Control Interface (RCI) and the Compliant Control Architecture (CCA) can deal with this platform from a software modeling and software architectural perspective. We focus on three different challenges: the first challenge is to enable reasonable and hierarchical *semantic abstractions* of the robot. The second challenge is to develop *hardware I/O abstractions* for the prototypical and heterogeneous technical setup. The third challenge is to realize this in a flexible and reusable manner. We evaluate our approaches to the above challenges in a practical scenario in which the robot is controlled either in simulation or on the real robot.

1 Introduction

Continuum robotic systems inspired by biological actuators like elephant trunks [1], octopus arms [2], or even squid tentacles [3] have gathered increasing interest in the last decade of robotics research. These systems move without traditional revolute or prismatic joints, but are based on continuous deformations in shape, and are typically driven by parallel hydraulic or pneumatic actuators. The focus of this paper is the Bionic Handling Assistant (BHA) [4] which is a new continuum platform inspired by elephant trunks and manufactured by *Festo* (see Fig. 1). The robot is pneumatically actuated and made almost completely out of polyamide which makes it very flexible and lightweight. The robot comprises three main segments, each with three parallel, pneumatic bellow actuators, a ball-joint as wrist, also actuated by three actuators, and a three finger gripper actuated by one bellow actuator. When the bellow actuators are supplied with pressure, they extend their length and can cause arc-like deformations as well as elongations.

The continuous arc-like deformations are not only challenging from a simulation and control point of view. On a semantic level, the multi-segment parallel actuation demands for software modeling approaches that allow an intrinsically hierarchical view on a robot. This is not the case for standard revolute joint robots which are appropriately modeled by a series of single-actuator abstractions. On a hardware level, the prototypical robot setup provides very heterogeneous I/O channels, including pressure sensing

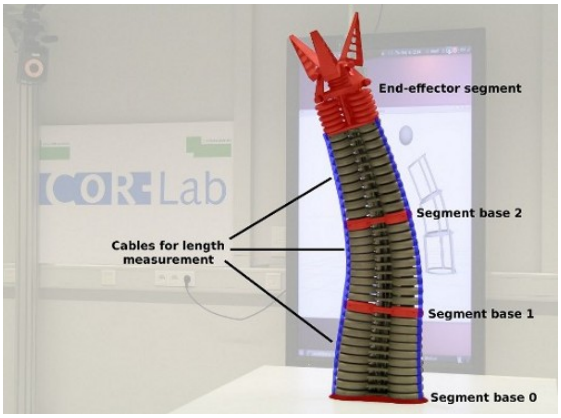

Fig. 1: The kinematic structure of the BHA comprises three main segments, each consisting of three parallel pneumatic bellow actuators. The length of these actuators can be determined with cable-potentiometers.

and control via a CAN bus, length sensing via an analog-digital converter PCI card, as well as position sensing with Vicon [5] motion tracking system that communicates with a proprietary network protocol.

This paper investigates the use of the software concepts and abstractions of the existing software frameworks Robot Control Interface (RCI) and the Compliant Control Architecture (CCA) to master these challenges in a *flexible and reusable* manner. Our contribution is extracting software requirements imposed by the above challenges of continuum robots, elaborated in Section 2. We then discuss these requirements along implementation of a length control use-case with RCI and CCA. Section 3 introduces RCI and shows how we mapped the BHA to its semantic abstractions. In Section 4 we highlight certain design decisions of the technology mapping and Section 5 shows a practical use-case, introducing our open-source BHA simulator as well as length control on the real robot. In Section 6 we conclude and point out the key aspects we identified to cope with the challenges of continuum robots like Festo‘s Bionic Handling Assistant.

2 Challenges

With all its advantages and properties desirable for physical Human-Robot Interaction (*pHRI*), the continuum kinematics of the BHA impose a number of challenges both on a semantical and a technological level. On the lowest level, the robot is driven by 13 bellow actuators. Each of them allows to control and sense pressure. Although there are physical interactions between these actuators they can be seen as independent and conceptually identical from a software point of view – similar to different revolute joints on standard robots. The situation changes when postural sensing and control is considered. The pressure in the actuators is not a reliable means to define a robot posture, i.e. a geometric shape of the robot, since it only describes a force. The BHA comprises

length sensors for all actuators except the gripper to circumvent this problem and provide geometric information. On the main segments these sensors are on the outside of each actuator. The length sensors of the wrist do not provide a "one to one" relation to the actuators since they are mounted in between two actuators. Although the main segments do have a "one to one" relation, the length of a single actuator is essentially meaningless: the parallel actuation design of three actuators in one segment only defines a shape when all three lengths come together. Hence, our approach is to use an entire segment as the elementary *semantic abstraction* that must be considered to describe the geometric configuration of the robot, which contradicts common ways of semantic and technical modeling of robotics platforms that focus on single actuator abstractions [6]. A software model must allow to work on top of this semantic abstraction as well as inside the segment abstraction in a hierarchical fashion: controlling the length is not done with standard PID control concepts and, in fact, the robot comes without a length controller. Solving the length control is active research and requires a software model that allows to plug-in control functionality into a segment abstraction that already provides sensing capabilities. Also the cartesian control of the end-effector is not available as standard method or component, although sensing is directly available.

A second challenge arises from the interfacing of the actual hardware I/O capabilities of the robot setup. The very heterogeneous hardware setup involves a number of different communication patterns, protocols and transports necessary in order to access and integrate all technical components of the robot. The different hardware interfaces use different data representations, have different timing and timing constraints, as well as completely different data volumes to process. The pressure control and sensing is interfaced with a custom binary protocol over a CAN-bus which is accessed with a Linux SocketCAN driver. This interface needs to be operated in (soft) real-time since it represents the instantaneous actuation of robot. The current measurements, commands, and possible error- or status messages need to be updated with a request-reply pattern at a rate of $50\,Hz$. Thereby the 13 actuators are controlled by two valve-units, which are connected to the CAN-bus as separate devices. Each valve-unit has eight valves, whereas the actuators of the lowest segment are each connected to two separate valves, both of which need to be actively controlled. The length measurement is interfaced with a proprietary driver accessing an analog-digital converter PCI card. The device allows for an almost instantaneous reading of the current lengths. Finally, the cartesian position of the end-effector can be sensed, but not controlled with a Vicon [5] motion tracking system. This system communicates with $200\,Hz$ via a proprietary network protocol. These heterogeneous channels need to be synchronized and leveraged in a coherent software framework which demands for *hardware I/O abstractions* that can capture the diversity of these devices.

The development of *flexible and reusable* abstractions for such components is a natural requirement for a robotics framework, although in particular the hierarchical modeling on a semantic level is not trivially achieved. The third challenge in our setup, however, goes beyond the reuse of segment- or whole-robot-abstractions: Current research on this very prototypical robot setup is mostly concerned with the development of controllers that mediate between high-level semantic abstractions and low-level hardware abstractions. This demands a high degree of flexibility and extendability even between

reusable abstractions on several levels. Abstractions that only allow sensing in the first place must be extendable by control-semantics.

The use case investigated in this paper is the control of actuator lengths. Length control is fundamental to any other application with the robot, but requires a solution to all of the above challenges: It has to

1. rely on hierarchical abstractions of robot segments instead of single actuators.
2. incorporate heterogeneous sensor information and feed commands back in realtime.
3. integrate in a flexible, additive manner into other reusable abstractions.

Thereby the BHA particularly challenges the classical conceptualizations of control interfaces, often used in robotics frameworks. Length control on the BHA for example can not be expressed in the classical length control interface, providing length measurement and its control at the same place. Whereas length is *measured* in the nine chambers, *controlling* of the length only makes sense in the context of an entire segment due to the strong coupling inside a segment.

3 Software Abstractions and Programming Model

In order to access to the platform-specific features of a robot platform in a generic and coherent way, we developed a framework of software abstractions for compliant robots, called Robot Control Interface (RCI) [7]. RCI provides a set of domain-specific abstractions to represent common features of compliant robotics systems. The domain-specific abstractions of RCI are imposed through the domain of motion learning on soft and compliant robots, whereas a *domain* is a *"set of current and future applications which share a set of common capabilities and data"* [8]. The representations are available as part of the Robot Control Interface and comprise software solutions as well as interfaces, features and domain objects identified in a domain analysis process. To be able to describe the domain in a condensed manner, we performed an extensive feature-oriented domain analysis (FODA [8]), including currently existing applications, robot-features, and software frameworks in the domain. Examples of applications and frameworks included in the domain analysis are: The *KUKA Fast Research Interface* [9], the *Orocos-RSI* extensions of the Orocos Framework [10], the RobotCub software [11], especially the recently developed *iDyn* library, as well as the two ROS stacks *force-torque* by the Healthcare Robotics Lab at Georgia Tech [12] and *COB force-torque* for the the Care-O-bot platform [13]. In addition to analyzing features of frameworks and interfaces, we analyzed interfaces of compliant robot platforms like *COMAN*, the compliant successor of *iCub* and the compliant quadruped robot *Oncilla*.

The unifying Robot Control Interface (RCI) follows a *model-driven approach* and abstracts from a concrete hardware platform while allowing to integrate or generate the necessary platform-specific code. The description of the logical architecture and API views yields a first definition of the interfaces between the robot platform (simulated or hardware) and the user application. RCI focuses on the software interface for *proprioceptive sensors and actuators*, which includes velocities, accelerations and forces applied to a robot as well as sensorics for equilibrium or balance (accelerometer, gyroscope). This API defines a low-level robot programming interface specifically considering the requirements of compliant actuator control and proprioceptive sensing.

A first observation as result of the domain analysis is that in this domain a clear distinction of physical robot parts between either sensor or actuator is often infeasible and sometimes even cannot be made. Instead, actuator and sensor parts share a comprehensive set of controlling and sensing features (i.e. sensing of position, force). This is valid for example for almost every actuator with position control, being able to sense at least the current encoder value, often also the motor current et cetera. In the domain of compliant robots, e.g. including active compliant actuators, a single actuator often provides a rich subset of features from actuators and sensors, including measurement and control of force, torque and joint position.

Based on this observation RCI defines a set of possible features of physical robot parts, both controlling and sensing features. We then define the *ResourceNode* as a logical abstraction for sensors *and* actuators, which can have an arbitrary set of these features. A ResourceNode with just sensing features represents a pure sensing robot part, a ResourceNode with controlling and optionally sensing features represents an actuated robot parts. RCI defines a number of fine-grained interfaces of semantic data abstraction (e.g. joint angles, torques, forces, et cetera) and a separate control and sensing interface for each. A core aspect of this fine-grained assignment of features to robot parts is, that the control and sensing aspect of the same variable (e.g. joint angle) can even be split to several robot parts.

A second essential concept in RCI is the concept of a *Synchronizer* that solves the connection of the robot‘s software abstractions (ResourceNodes) with the actual robot – in simulation or hardware. The synchronizer reads commands sent to the ResourceNodes and passes them as commands to the robot. It also reads sensor values from the robot and passes them to the Resource Nodes.

The Robot Control Interface provides *platform-specific* software *abstractions* for modeling of the robot platform, in the form of the ResourceNode abstraction for sensors and actuators as well as a set of fine-grained controlling and sensing features. It also provides the *implementation-specific* abstraction of hardware I/O interfaces in the form of Synchronizers. This set of abstractions provides the *Programming Model* for implementations of a robot interface with RCI.

3.1 Modeling of the Bionic Handling Assistant

In order to check whether the Programming Model provided by the Robot Control Interface is capable of meeting the challenges depicted in Section 2, we modeled the BHA with RCI abstractions. We evaluate whether the concept of RCI ResourceNodes can provide reasonable *semantic abstractions* of the continuum kinematics and therefore provide a solution to the semantic challenges. Furthermore we evaluate whether the RCI concept of a Synchronizer is able to cope with the heterogeneous hardware interface setup of the BHA and provides meaningful *hardware I/O abstractions*.

Note, that for the sake of clarity in this example we focus on the first nine, main bellow actuators without wrist and gripper actuation.

Modeling of the platform-specific parts of the robot system, the semantic abstractions, is done in three kinds of different resource nodes: i) chambers, ii) segments and iii) the end-effector.

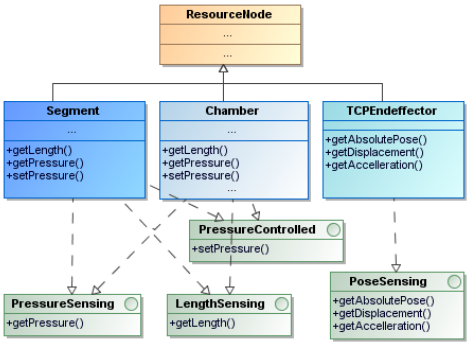

Fig. 2: RCI ResourceNodes for the BHA's three main segments, its chambers and the end-effector.

- A **chamber node** represents a bellow actuation unit of the robot, equipped with length sensing. Interfaces of this node are therefore `PressureSensing`, `PressureControlled` and `LengthSensing`.
- A **segment node** does not add any functionality at this point, but repeats the length sensed values and the pressures of its three chambers. The segments provide the three chamber length values in a semantically coherent way in order to provide a basis for later-on extensions with control capabilities.
- The **end-effector node** is the gripper, which cartesian position is sensed. Although the position is sensed by an external component, we model it as part of the robot system, since in our context (cf. use-case in Section 5) this is a relevant part of making the robot system usable.

The implementation-specific part of the robot system is modeled as set of synchronizers that have to deal with the diverse hardware interfaces for pressure control and sensing, length sensing and the Vicon system.

1. The **PressureSynchronizer** connects over the CAN bus to receive pressure values, writes them to the chamber nodes. It reads pressure commands from the chamber resource nodes and sends them via CAN to the responsible valves-unit.
2. The **LengthSynchronizer** accesses the driver of the analog-digital converter PCI card for reading values of the cable potentiometers at the outside of the bellow actuators, and writes them to the chamber resource nodes.
3. The **ViconSynchronizer** runs on a different workstation. It connects to the Vicon motion tracking server and reads the current end-effector position which are set as current sensor value in the end-effector node.

4 Technology Mapping

The following section highlights our design decisions for the technology mapping that we use in order to prove the above abstractions are suitable for applications on the BHA.

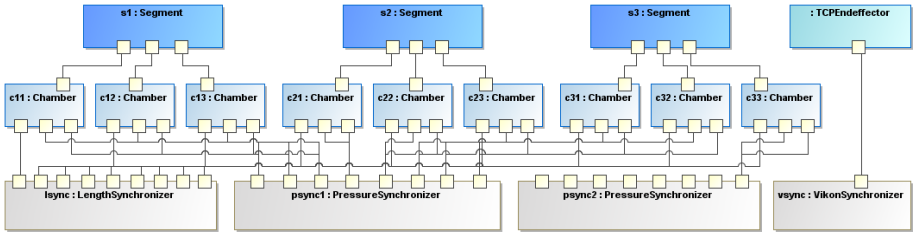

Fig. 3: Organization of BHA Resource Node instances (blue) and Synchronizer instances (gray). Note that there are two pressure synchronizers, one for each of the valve units. The first eight chambers connect to the first valve unit, the ninth chamber is connected to the second valve unit.

The technology mapping includes the implementation of the abstractions in the Robot Control Interface, a component framework for extending the interface, applications running on the robot, as well as the middleware for integration and access to the hardware interfaces.

4.1 Robot Control Interface

Robot Control Interface is available as C++ library `librci`, providing interfaces and base implementations for the abstractions introduced in Section 3. The library includes a set of domain-specific base interfaces (various `...Controlled` and `...Sensing` interfaces, as well as generic implementations of its setters and getters. Features that allow to apply commands to a node, are expressed in the `...Controlled` interfaces, allowing to set a reference for a certain controller (position, length, pressure, etc.). The set of `Controlled` interfaces implemented collectively by all nodes of a robot defines a set of tasks that can be executed by the robot. Features that allow getting status information from a node, are expressed in the `...Sensing` interfaces. The set of `Sensing` interfaces implemented by nodes of a robot defines a set of status data that can be reported by the robot.

Additionally the RCI library provides a collection of domain-types serving as data-holders with domain-specific manipulation methods, setters and getters. The library will soon be open-source as part of the European AMARSi project.

4.2 Component Architecture

For implementation of functional components and integrating RCI entities into the applications, we leverage the Compliant Control Architecture (CCA). CCA is an event-based, middleware-agnostic component architecture for robotics research, focusing on (real-time) control of compliant hardware and enabling machine learning. The C++ library `libcca` serves as a technology mapping for platforms modeled in RCI and as

component architecture for implementing user applications and the adaption of hardware and simulation interface as later-on described in Section 5.2. An application implemented using CCA is a graph of loosely coupled components which extend the CCA base component. CCA components possess ports to exchange data between each other via data-flow. Data in this data-flow graph is represented by domain-types, which specify the representation of data and provide domain-specific access methods. Data-flow in CCA can express acyclic as well as cyclic messaging graphs.

For management and coordinated processing of rich component graphs, CCA provides the concept of *ProcessingStrategies*. These strategies allow for a loose, input-driven coupling of nodes, but also for control-loops with tight timing constraints. CCA comes with an extendable set of basic strategies, like *TimedProcessing*, which processes a component based on a fixed timing, and *PortTriggered*, which processes a component triggered by incoming data at a specific input port. Components using timing-based processing strategies can also be executed in a real-time context, based on the Xenomai Real-Time API. By combining collections of real-time-based components, entire subgraphs can be executed in real-time context. To ease the construction of subgraphs, CCA provides a set of static connectivity skeletons (often called *algorithmic skeletons* in parallel computing literature), like pipelines, splitters and collectors. A pipeline is a set of components with input-driven processing, which processes the entire data as fast as possible through the sequence of connected components. A splitter takes a multi-dimensional data-items on its input port and splits it into sub-elements. A collector is collecting and merging incoming low-dimensional data-items to an outgoing multi-dimensional data item.

4.3 Middleware and Integration

In order to realize the data-flow between CCA components we facilitate the open-source robotics middleware *Robotics Service Bus* (RSB [14]). RSB is a lightweight, event-based, and highly customizable robotics middleware with native implementations in C++, Java, Python and Common Lisp.

Robotics applications often involve computationally complex modules, especially applications involving machine learning or complex kinematics modules (e.g. inverse kinematics for multiple limbs, continuum kinematics). RSB offers local and remote transports, that can transparently be switched during runtime, allowing CCA components to be distributed at runtime over several machines, for example the robot itself, workstations, as well as computation clusters. CCA components on the same machine communicate in a fast local inprocess transport, and only communicate with a network stack if they are distributed over different other workstations. Since type of communication (remote, local) is configured at runtime, a CCA component graph can be deployed in an arbitrary configuration across different computing devices.

In this sense RSB also provides an answer to the technical challenges. All involved hardware interfaces are connected to the system and therefore available for all software components through the middleware. End-effector positions are streamed from the VICON control PC to the functional components, pressure and length values are bridged to the CAN bus and IO card respectively.

5 Use Case

In this section we validate the above concepts, that were already proved working on robot platforms like iCub, Oncilla and KUKA LWR IV, on the continuum robot platform BHA. This provides a first measurement of the conformance of our suggested API and architecture to compliant robotics and successfully integrated our concepts on state-of-the-art hardware and simulated robots from the rigid body domain.

A first use-case for working with the BHA is length control. For a reliable positioning, it is not sufficient to control the pressure alone: Friction, hysteresis and non-stationarities can cause largely different postures on the BHA when supplying the same pressure several times. In particular during dynamic movements the pressure is not sufficient to determine the posture or position of the robot, since it only expresses a force on the actuators. Length control is therefore a central skill for any other task to be achieved on the robot.

5.1 Length Control on the real platform

A naive approach to the control of the actuator lengths would be to consider each actuator in isolation, and adjust the pressure with a standard PID feedback controller until the desired length is achieved. From a pure control perspective, this approach has two severe disadvantages:

1. Feedback control can only be done with very low gains on a pneumatic robot due to long temporal delays in the actuation chain, which corresponds to slow motion.
2. It does not consider the strong mechanical interplay of actuators in the same segment, which largely influences the pressure necessary to achieve a certain length.

In order to deal with both problems at the same time we use machine learning methods and estimate models that allow a feedforward control of an entire segment. A length command, generated by some application, is smoothed by a generic filtering mechanism. This three-dimensional length command is fed both to the learned model and a

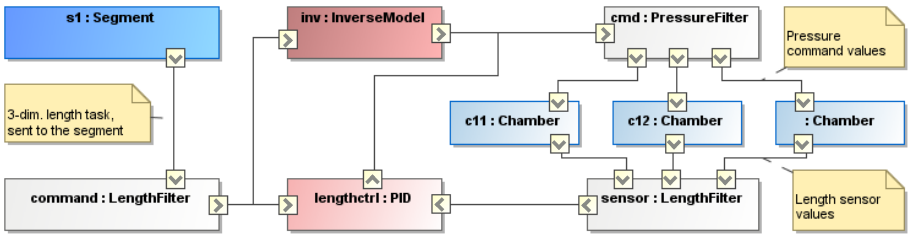

Fig. 4: Length control of the Bionic Handling Assistant. For the sake of clarity illustrated for the first of the three segments only. Resource nodes (segment, chambers) in blue, filters (length, pressure) in grey, and control components in red.

generic PID controller, which receives Kalman-filtered feedback about the current actual lengths. Both controllers output a pressure signal. These signals are added, filtered, and supplied to the segments as new target pressure.

This length control scenario is realized with a set of – generic as well as rather platform- and application-specific – CCA components as illustrated in Figure 4. The chamber nodes are domain-specific implementations for the BHA, while PID-controllers and various filters are generic components that are reused from other domains. Pressure and length data passed between the components are the domain-types defined by the Robot Control Interface. The inverse model is a clear platform- and application-specific component. The integration of this length control scheme uses the two essential properties of our modeling approach:

1. It uses the hierarchical modeling of segments and chambers and thereby utilizes the semantical grouping of sensory and command values in a segment abstraction.
2. It adds a `LengthControlled` interface to the segment, which was previously only a `LengthSensing` abstraction and an aggregator for the chamber values. This interface allows to set three-dimensional target values for the lengths.

Thereby the very fine-grained architectural design facilitates the reuse on component level which we extensively exploit by using filtering and feedback-control implementations that were developed in other scenarios.

5.2 Common Interface for Simulation and Hardware

The hierarchical aggregation of lengths into a segment abstraction is even more important when pure simulation is considered as alternative to the real robot scenario. We use an open source implementation [15] of a continuum kinematics model in order to model the kinematic structure of the BHA. This model assumes that bending and stretching movements of each robot segment behave like a torus section (see Fig. 5) which allows to infer the coordinate transformations for the forward kinematics. The model allows to predict the end-effector position of the BHA based on the actuator lengths with an accuracy of $1\,\%$ relative to the robot size [16].

The control of lengths in this scenario is trivial, because they can directly be "set". A substantial difference to the real-robot scenario is that single actuators do not exist. From a mathematical modeling point of view a single actuator is not a valid concept since it does not allow to determine any component of the six-dimensional pose-transformation involved in one segment. The entire simulation operates on aggregates of three lengths.

This simulation scenario can directly be described with the existing Robot Control Interface infrastructure and the abstractions developed for the real-robot as presented in the previous sections. The segment abstractions are entirely reused from the real-robot implementation, chambers and controllers are simply left out. The only difference in implementation is that a new `Synchronizer` is needed in order to connect segments and simulation software. Since the segment abstractions entirely decouple the backend-implementation (`Synchronizers`) from the application code, it is directly possible for an application to switch between real-robot and simulation, which is even possible during runtime.

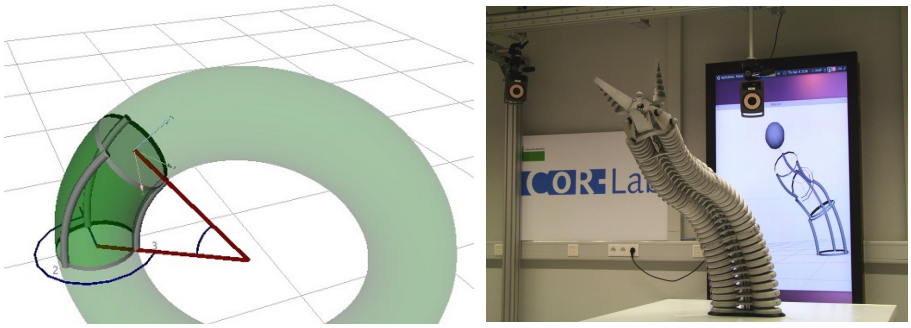

Fig. 5: Torus model to simulate BHA's kinematic structure (left) and on the basis the 3D visualization of the BHA (right).

6 Conclusion

In this paper we discussed the software challenges imposed by continuum robots, based on our exemplary platform, Festo's Bionic Handling Assistant. An analytical approach to find fine-grained software abstractions based on a feature-oriented domain analysis and a resulting programming model for robot interfaces were introduced. In the course of the paper we discussed these software abstractions and our technology mapping, the Robot Control Interface and the Compliant Control Architecture, along the practical use-case of length control on the real robot platform and simulation of the Bionic Handling Assistant. The contribution of this paper, the software abstractions, especially the clear separation of control and sensing aspects, and the hierarchical modeling of segments and chambers, showed to be helpful and necessary to master these challenges in a *flexible and reusable* manner.

The length-control use-case presented in this paper is fundamental to any application on the robot and has already served as basis for learning of reaching skills [17]. Similar to length-control, this learning adds a `Controllable` concept to a previously only sensible effector position, and has already been exploited in several applications.

Acknowledgements

The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 – Challenge 2 – Cognitive Systems, Interaction, Robotics – under grant agreement No 248311 - AMARSi.

References

1. M. W. Hannan and I. D. Walker. Kinematics and the Implementation of an Elephant's Trunk Manipulator and Other Continuum Style Robots. *Journal of Robotic Systems*, 20(2):45–63, 2003.
2. C. Laschi, B. Mazzolai, V. Mattoli, M. Cianchetti, and P. Dario. Design of a Biomimetic Robotic Octopus Arm. *Bioinspiration & Biomimetics*, 4(1), 2009.
3. J. F. Wilson, D. Li, Z. Chen, and R. T. George. Flexible Robot Manipulators and Grippers: Relatives of Elephant Trunks and Squid Tentacles. In *Robots and Biological Systems: Towards a New Bionics*, volume 102 of *NATO ASI*, pages 475–494. 1993.
4. A. Grzesiak, R. Becker, and A. Verl. The Bionic Handling Assistant: A Success Story of Additive Manufacturing. *Assembly Automation*, 31(4):329 – 333, 2011.
5. VICON. Motion Tracking Systems. http://www.vicon.com.
6. D. Branson, R. Kang, E. Guglielmino, and D. G. Caldwell. Control Architecture for Robots with Continuum Arms Inspired by Octopus vulgaris Neurophysiology. In *International Conference on Robotics and Automation*, pages 5283–5288, 2012.
7. A. Nordmann, S. Wrede, N. Tsagarakis, and A. Tuleu. Software Interface for Proprioceptive Sensors and Actuators. Technical report, AMARSi, 2010. http://www.amarsi-project.eu/system/files/AMARSI-D.7.1.pdf.
8. S. G. Cohen, J. A. Hess, W. E. Novak, and A. S. Peterson. FODA: Feature-Oriented Domain Analysis. 1990.
9. KUKA. Fast Research Interface - Preliminary Documentation. Technical report, 2010.
10. The Orocos Project. Open Robot Control Software. http://www.orocos.org/.
11. P. Fitzpatrick, G. Metta, and L. Natale. Towards Long-lived Robot Genes. *Robotics and Autonomous Systems*, 56(1):29–45, 2008.
12. Healthcare Robotics Lab Georgia Tech. force-torque Package. http://www.ros.org/wiki/force_torque.
13. A. Bubeck. Care-O-bot force-torque Package. http://www.ros.org/wiki/cob_forcetorque.
14. J. Wienke and S. Wrede. A Middleware for Collaborative Research in Experimental Robotics. In *International Symposium on System Integration*, pages 1183–1190, Kyoto, 2011.
15. Research Institute for Cognition and Robotics. Software: Continuum Kinematics Simulation. https://www.cor-lab.org/software-continuum-kinematics-simulation.
16. M. Rolf and J. J. Steil. Constant Curvature Continuum Kinematics as Fast Approximate Model for the Bionic Handling Assistant. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012.
17. M. Rolf and J. J. Steil. Efficient Exploratory Learning of Inverse Kinematics on a Bionic Elephant Trunk. *IEEE Transactions on Neural Networks and Learning Systems*, submitted, 2012.